A New Approach in Building a Corpus for Natural Language Generation Systems

Mª del Socorro Bernardos Galindo[1] and Guadalupe Aguado de Cea[2]

[1] Laboratorio de Inteligencia Artificial,
Facultad de Informática, Universidad Politécnica de Madrid.
Campus de Montegancedo, s/n. 28660 Boadilla del Monte, Spain.
sgalindo@delicias.dia.fi.upm.es

[2] Departamento de Lingüística Aplicada a la Ciencia y a la Computación,
Facultad de Informática, Universidad Politécnica de Madrid.
Campus de Montegancedo, s/n. 28660 Boadilla del Monte, Spain.
lupe@fi.upm.es

Abstract. One of the main difficulties in building NLG systems is to produce a good requirement specification. A way to face this problem is by using a corpus to show the client the features of the system to be developed. In this paper we describe a method to elaborate that corpus and how can be used for a particular system. This method consists of five steps: text collection, input determination, text and input analysis, corpus construction, and pattern extraction.

1 Introduction

The development of an NLG system involves the performance of multiple tasks, such as the study of the linguistic phenomena to be handled, the design of its architecture, the construction of a grammar, to mention just a few. One of the main tasks (and among the first to face) is to make a requirements analysis in order to yield an initial specification of the requirements of the system. There are general techniques, applied in software engineering [13], that can be used here, but using specific methods for NLG can be much more interesting. One of the most representative ones is described in [14], where the developer uses a corpus to describe the user the system under construction. Its advantage is that it is much easier to talk about the functionality of the system with the users showing them examples than speaking about NLG in a more abstract way, especially because many users do not have previous experience with NLG technology. In this paper a method to elaborate a corpus based on that approach is introduced.

Before describing the method, it is convenient to explain what is understood by the term "corpus". According to [6] a corpus is a "closed set of texts established according to specific structural criteria" It can be considered as the result of an empirical process of linguistic data collection combined with the author's own capacity of elicitation and introspection. Taking this definition as a base, and adding some nuances specific to the NLG field (taken from [5]), a corpus can be defined as

A. Gelbukh (Ed.): CICLing 2001, LNCS 2004, pp. 216-225, 2001.

"a set of examples of output texts and input data that specify the functionality of the NLG system to be developed".

The reason to include the input data is that it is senseless to have texts in the corpus that will never be generated, i. e. corresponding to no input.

The relevance of the corpus does not lie only in its usefulness to specify the requirements of the systems, but also in its use as a resource to study the linguistic phenomena the system has to handle.

2 Method

The process to elaborate a corpus for an NLG system presented here consists of five steps: output text collection, input determination, text and input analysis, corpus construction and pattern extraction. The first four steps help to build the corpus. We start by collecting a set of possible output texts and input data. Then an analysis of both sets is made and finally, that analysis is used to construct the corpus, where each output text has an input associated. The last step, pattern extraction, is useful only if the reuse of the work is intended.

2.1 Output Text Collection

In this phase, the aim is to obtain a set of texts, belonging to the application domain, that cover the complete range of the texts the system is expected to produce. We can use several sources to get this set of texts: reference books, experts in the domain, and documents or reports written in the past.

It is important in this stage to have gathered a great number of texts. Their final usefulness is less interesting than to have a sample as complete as possible of all the possibilities that can be presented as output text.

2.2 Input Determination

The goal of this step is to determine the kind of inputs the NLG system has to face.

Based on our experience, the input of an NLG system has been identified by four components: the communicative goal to achieve, the model of the user to whom the text is addressed, the history of the discourse that has already taken place, and the user request.

A *communicative goal* can be defined as what one intends to express and what it is for. Communicative goals direct the generation process: they determine what should be selected from the information source, how the text should be structured, what words should be used, etc. These goals can have different complexity levels, some are simple, such as providing information about some contents of the knowledge or data base, but others are quite complex, such as trying to persuade someone to do something.

The *user model* is a characterization of the recipient (reader) of the text. Among the several aspects than can be included in a user model are: the information about the

user's expertise level, his/her preferences and the task he/she intends to do. This enables the customization of the text.

The *discourse history* is a model of all the previous interactions between the user and the system. It is used so that the output text can be generated counting on what has already been discussed on previous occasions.

The *user request* is what the user enters in the system in a particular interaction. It can vary widely depending on the sort of system. For instance, in a document generation system, it would be the request for a particular document (such as the meteorological report of a day, a week or a month, etc.); in an information retrieval system, the concrete query, for example.

All these components are not necessarily present in every system. There can be systems that do not care about the user, or do not mind repeating what has already been said, or they always have the same goal, etc.

Unlike in the text collection step, we do not use concrete data, specially because the amount of user requests can be huge. Here we are interested in the possible kinds of requests. It will be later, when we associate the input data to their corresponding output text (see Sec. 2.4), when we will need the concrete combination of the different components of the input and also their concrete value.

This phase, input determination, can be carried out previously or simultaneously to the text collection.

2.3 Text and Input Analysis

The aim of this phase is to get a detailed understanding of the correspondences between the output texts and the data of the available information source (data base, knowledge base, etc.), and between the output texts and the input data.

The first thing can be achieved analysing the content of the texts. To do so, the clauses of the text can be classified into the categories proposed by Reiter and Dale [14]:

- *Unchanging text*: A textual fragment that is always present in the output texts.
- *Directly-available data*: Text that presents information that is available directly in the input data or an associated information source.
- *Computable data*: Text that presents information that can be derived from the input data via some computation or reasoning.
- *Unavailable data*: Text that presents information that is not present in or derivable from the input data.

This last kind of texts cause the most problems, since the information needed to generate them can not be obtained. This can be solved by making more information available to the system, changing the text to eliminate the affected parts, and expecting the human to write such text.

As for the second thing, the study of the relations between the output texts and the inputs would be useful to establish not only the correspondences between both of them, but also to find out those texts without an associated input and vice-versa.

Finally, the form of the texts has to be analyzed too. It is necessary to investigate if for a given input there are several texts with the same content and very different style (e. g., because they belong to different experts) and to study what these styles are like.

In addition, the non-optimal texts (e. g., because they are difficult to understand) and the ones that can be improved have to be detected.

2.4 Corpus Construction

Based on the analysis done in the previous phase, and the criteria established by the clients and the developers, it is time to determine the corpus in which the texts the NLG system will be able to generate and the inputs related to them are included.

There are several tasks that can be done in this process:

- Rejecting or modifying the texts that refer to unavailable data when the solutions mentioned above are technically impossible or prohibitively expensive to adopt any of them.
- Simplifying texts that are difficult or impossible to generate.
- Adding inputs to the texts that do not correspond to any input or removing them.
- Adding output texts to the inputs that do not correspond to any text or removing them.
- Modifying texts that are difficult to understand or can be improved.
- Solving conflicts, as the ones caused by different experts.

For a better result, it is advisable to make the corpus in close collaboration with the experts (linguistic and domain ones) and the clients, each contribution should take place at the appropriate moment. Experts will ensure that it is correct (both grammatically and in the domain) and discussion with the clients will be useful to take decisions about eliminating, including or modifying an input or a text.

Quite often, the corpus obtained will not be the final one, because the little experience in the NLG field makes it difficult to establish from the beginning what can be done, i.e., what linguistic phenomena will be handled.

2.5 Pattern Extraction

After the corpus is elaborated, the patterns of the texts to be generated can be designed.

First of all, the structures of the texts have to be determined replacing the domain words with general ones, such as concept, property, etc. The resulting texts can be thought of as the "skeletons" of the texts. Next, a notation to represent the patterns has to be decided. Finally the different skeletons are united using the notation.

This phase is useful only in case the system is intended to be as independent of the application domain as possible, so that it can be reused. In these situations the use of patterns as the base of generation will be useful so that text with the same structure, even belonging to different domains, can be generated correctly.

The idea is that the patterns can help the developers to see if there are similar texts generated for other projects. Once the corpus is finished they can examine the patterns produced in other projects and study if they can be applied to the texts of their project. After that, they can reuse the previous results for similar texts whenever it is possible and convenient. In this way, time and effort can be saved.

As we have already said, the corpus is very likely to suffer from continuous modifications throughout the development of the system. In those cases in which the patterns have been extracted, the most advisable thing is to make the modifications in the patterns. In this way, besides keeping the intended independence, some repetitive work is avoided: the modification of the texts and the reflection of those changes in the patterns.

3 Elaboration of a Corpus in the ONTOGENERATION Project

The method described above was applied in the ONTOGENERATION project [1] but adapting it to the specific situation. The aspects that had more influence in the corpus were:

- The information source was predetermined, namely, the *Chemicals* ontology [7]. This fact caused the range of texts to be quite limited and even in the first moments a lot of useless material had to be rejected.
- The result was intended to be reused with other information sources, basically other ontologies.

Let us look at how the process to elaborate the corpus was carried out.

3.1 Text Collection

First of all, several books on Chemistry, such as [8][10], were examined in order to collect the texts referring to the chemical elements and their properties.

But these texts were considered to be very long and they included a lot of information that was not, and would never be[1], present in the *Chemicals* ontology. That was the reason why it was decided not to use those texts as they were but rather eliminate some parts and create new ones from others instead[2].

Each member of the team prepared a group of possible texts and then a new set was created jointly from those groups. Since the texts were not exactly those appearing in the books, an expert was consulted to validate its suitability and correctness in the chemical domain.

3.2 Input Determination

Once the text collection was finished, the following input was determined, taking into account the aspects mentioned in 2.2.:

- *Communicative goal*: The system had a single aim: to provide all the information requested by the user. Since there was only one, there was no need to make it explicit.

---

[1] The contents of these texts were so clearly out of the requirements of *Chemicals* that it was out of discussion their inclusion in the ontology.

[2] This is a very superficial purge for which it is not necessary a detailed analysis (carried out later), it just helps to save time in posterior phases.

- *User model*: There were no variations in the texts due to the differences between the users.
- *Discourse history*: Every time a new query was made, it was supposed to be isolated from the others, so there was not a real discourse history.
- *User request*: This component consisted of all the queries the user could make to the NLG system. These queries could be about anything included in the *Chemicals* ontology (concepts, instances, attributes, axioms and functions).

As the reader can deduce, the input to the generation system had only one component, the query made by the user. This part was studied more deeply in this step, so that different kinds of queries could be determined.

First, the queries were divided into five categories:

- Chemical group descriptions
- Formula descriptions
- Chemical element descriptions
- Chemical element comparisons
- Chemical element property descriptions

These categories were refined later and we got a total of thirteen types classified according to them. For example: the class "chemical group description" included "concept classification", "concept definition", concept decomposition", "concept properties", "concept examples" and "concept synonyms".

The concrete queries were not determined until the correspondences between the output texts and the inputs were established.

3.3 Input and Text Analysis

In this phase, there was a discussion about which of the collected texts should be included as an answer to each kind of query. Each text could be associated to more than one query. Thus, the sentence "Non transition metals are divided in alkalines and alkalineterreus", corresponded to concept classification, concept definition and concept decomposition.

On the other hand, the analysis of the content of the texts showed that, according to Reiter and Dale's classification [14], the texts showed the following characteristics:

- Unchanging text: Complete utterances were not always present in the same way.
- Directly available and computational data: No computations are needed except for those expressions that refer to information stored in the description of the ontology terms, i. e. those containing data that are not related to chemical properties as in "Argon was discovered by Lord Rayleigh and Sir William Ramsay in 1894" or "Arsenium is poisonous

Note that the texts that contain these clauses are not included as an answer to any of the inputs determined previously.

- Unavailable data: There are texts like "Reactiveless elements are classified according to its metallic feature, thus, there are non metals, semi-metals and

metals", that are impossible to generate because the ontology does not have concept attributes.
Other unavailable data were detected in clauses that expressed the qualitative value of a property as in "The electronegativity of the bromum is high". This is because *Chemicals* does not provide data useful to state if a value is high, medium or low.

There was nothing special to report on the form of the texts.

3.4 Corpus Construction

The first corpus that was built had the texts collected in the first step, classified according to the kinds of queries determined in the second step, and also based on the analysis of the third step, in which the following clauses were eliminated:

- Those that referred to concept attributes. The inclusion of concept attributes clashed with the requirements of *Chemicals*, so we decided to remove those texts referring to that kind of attributes.
- Those that had computable data related to the description field. Several options were studied for this situation; e. g., canned text and text analysis. The first was impossible since the description was in English and our system generated text in Spanish. The effort involved for the second did not make it advisable.

According to the analysis made, clauses expressing qualitative values could present some problems too. After studying the ontology and the different options, the decision was to include a "scale mechanism" that supported the generation of this kind of expressions.

3.5 Pattern Extraction

Since one of the goals of the project was that the final result was as source-independent as possible, patterns of the texts were created so that the sentences could be generalized to other information sources different from *Chemicals*.

Using the notation reflected in **Table 1**, 95 patterns were defined, most of them corresponding to more than one sentence. For instance[3]:

> *Concept-name {, concept-name}*$_0^n$ and *concept-name* [**make up** *article* |**are included in** *article* |**are called**] *concept-name.*

can refer to:

> "First transition series, second transition series, third transition series and actinides make up non transition metals."

and

> "Halogens are included in the metals class."

among many others.

[3] This pattern has been adapted from the real one, to make it simpler and understandable for non-Spanish speakers.

A → B	A has the same structure as B
[A]	Optional element
[A | B | C]	A or B or C, al least one of them
$\{A\}^{m}_{n}$	A is repeated between n and m times
A → B | C	A has the same structure as B or C
bold	fixed words
italics	variable words

Table 1. Pattern notation

3.6 Corpus Evolution

The first modification made to the corpus was organizational and it was due to the tool used for the development of the grammar[4] and the generation, namely KPML [3]. KPML only produces sentences, not paragraphs[5]. So patterns were classified in two groups: sentences and paragraphs. Paragraphs were correspondingly divided into sentences. The generation of those sentences can be thought of as an intermediate goal to get the target text.

Other changes were due to the relation between the texts and the queries. At the beginning a text could be associated to several queries. To simplify the planning of the texts, this situation was changed and a pattern could only correspond to a single query.

The last modifications consisted in a reduction of the corpus so that the project could be finished within the time established at the beginning of the project. We examined the corpus and selected three patters at most for each type of query.

It is worth pointing out that the modifications on the corpus were done in continuous interaction with the domain experts, so that they could state the things they disagreed with or could be improved.

To sum it up, the changes made in the initial corpus were due to:

- Design decisions.
- Expert corrections.
- Time planning.

All the changes were done directly on the patterns.

The final corpus contains 10 kinds of queries (classified in four groups) and 28 patterns. It can be seen in [12]; [4] and [11] show the corpus at different moments of the project.

[4] The grammar was developed according to Halliday's functional grammar [9].

[5] For the purpose of this project, we understand 'paragraph' as the units of written language that begin with a capital letter and end with a full stop. The next paragraph begins on a new line.

4 Conclusions and Future Work

A method to build a corpus in the area of NLG has been presented. This method systematizes the task of the analysis of the requirements related to the input data and the output texts of an NGL system in five steps: text collection, input determination, text and input analysis, corpus construction and pattern extraction. The result of the process (the corpus) is not only useful as the system requirements but also as a resource for the analysis and the design of the system.

The main contributions of the method described here are:

- There is a phase in which the input data are determined. These data are associated with the corresponding output texts in a later phase. In this way the corpus will never have texts the system will never generate. To help the developers to carry out this step, the components of the input have been identified: communicative goal, user model, discourse history and user request.
- There is a special emphasis on reusing software components and resources. The use of patterns can help to reuse the different products in other projects with similar features.

Each phase has been explained and some guidelines to perform them have been given. Note that the ideas expressed here are general and will have to be adapted to the particular application, as has been shown for the ONTOGENERATION project.

There remains to study to what extent this method can be applied to other NLG systems different from ONTOGENERATION, and see which modifications are needed. With that in view, other NLG projects are being considered. Those projects are intended to produce the same kind of texts as ONTOGENERATION to find out the real usefulness of the extracted patterns.

References

1. Aguado, G., Bañón, A., Bateman, J., Bernardos, S., Fernández, M., Gómez, A., Nieto, E., Olalla, A., Plaza, R., Sánchez. A.: Ontogeneration: Reusing Domain and Linguistic Ontologies for Spanish Text Generation. Workshop on Applications of Ontologies and Problem Solving Methods, ECAI'98, Brighton (1998)
2. Bañón, A.: Modelo de generación multisentencial EPRS. Trabajo Fin de Carrera, Facultad de Informática, Universidad Politécnica de Madrid, Madrid (1999)
3. Bateman, J. A.: Enabling Technology for multilingual natural language generation: the KPML development environment. In Natural Language Engineering, Vol. 1. Cambridge University Press, Cambridge (1997) 1-42
4. Bernardos, S.: GUME: Extensión de la Ontología GUM para el Español. Trabajo Fin de Carrera, Facultad de Informática, Universidad Politécnica de Madrid. Madrid (1997)
5. Dale, R. and Reiter, E.: Tutorial on Building applied Natural Language Generation Systems. ANLP-97 (1997)
6. Francis, W.N.: Language Corpora B.C. in J. Svartvik (ed) Directions in Corpus Linguistics. Proceedings of the Nobel Symposium 82. Berlin. Mouton de Gruyter (1992)
7. Fernández, M.: *Chemicals*: Una Ontología de Elementos Químicos, Trabajo Fin de Carrera, Facultad de Informática, Universidad Politécnica de Madrid (1996)
8. Fernández M. A. *et al:* Ciencias Naturales GAIA. Vincens-Vives. Barcelona. España (1995)

9. Halliday, M. A. M.: An Introduction to Functional Grammar. Edward Arnold, London (1985)
10. Lasheras, A. L. and Carretero, M.P. Física y Química. POSITRON Vincens-Vives. Barcelona. España (1987)
11. Nieto, E.: Metodología para adaptar una gramática sistémica-funcional para la generación en castellano. TFC, Facultad de Informática, Universidad Politécnica de Madrid (1999)
12. Olalla.A.: Sistema de GLN basado en ontologías: Ontogeneration. Trabajo Fin de Carrera, Facultad de Informática, Universidad Politécnica de Madrid (1999)
13. Pressman, R.: Software Engineering: A Practitioner's Approach. McGraw-Hill (1994)
14. Reiter E., and Dale, R.: Building Applied Natural language Generation Systems. Journal of Natural Language Engineering, 3(1). (1997). 57-87